Project Report

on

Data Analysis On Disaster

Submitted By:-
Sheetal
Nisha

Submitted in partial fulfillment of completion of the course

ADVANCE DIPLOMA IN IT NETWORKING & CLOUD COMPUTING

In

MINISTRY OF SKILL DEVELOPMENT ENTREPRENEURSHIP NSTI (W) NOIDA

APRIL-2023-2024

Abstract

the cake website is a digital heaven for cake enthusiast and connoisseurs alike. with delight blend of exquisite design declarable offering a seamless functionality, our platform aim to be the ultimate designation for all things cake-related .Through a visually capitative experience, user can Explorer derive array of cake varieties. learn about our passionate team and order customer creation with ease. The website not only certified creative but also provide valuable insights into cake Trends baking tips, and customer feedback. we invite you to include in our sweet word where every clicks Leads to a slice of happiness

ACKNOWLEDGEMENT

We take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. We extend our sincere and heartfelt thanks to our esteemed guide, Edunet Mentor **Mrs. Deepika Singh, Mrs. Mala Mishra and Ms. Ankita Shukla**, for providing us with the right guidance and advice at the crucial junctures and for showing me the right way. We also take this opportunity to express a deep sense of gratitude to our Trade faculty Staff Mrs. Ankita Shukla for their cordial support, course coordinator **Mr. D.A.Guruvulu**, valuable suggestions and guidance. We extend our sincere thanks to our respected Head of the division **Smt. Shashi Mathur** [JD NSTI(W) NOIDA], for allowing us to use the facilities available. We would like to thank the other faculty members also, at this occasion. Last but not the least, we would like to thank our friends and family for the support and encouragement they have given us during the course of our work.

Team Composition and Workload Division

1. Sheetal – Data Analysis & Synopses
2. Nisha -Data Analyses

Project Requirements

Project	Data Analysis On Disaster
Language Used	Python (Data Analyzation Data Visualization, sklearn)
Editor	Jupyter Notebook ,Google Colab
User Interface Design	HTML, Python
Web Browser	Google Chrome

Tables of Content

Introduction

The data analysis project focuses on comprehensively examining the occurrence of various types of disasters within a specific year and delving into their consequential impact. Through meticulous exploration of datasets encompassing disaster types, locations, and associated details, the project aims to uncover patterns, trends, and correlations. By scrutinizing factors such as casualties, damages, and response times, the analysis seeks to provide valuable insights into the dynamics of disasters during the designated timeframe. The overarching goal is to equip stakeholders with a nuanced understanding of the diverse challenges posed by different disasters, fostering informed decision-making, and enhancing disaster preparedness strategies.

Problem Statement

While the project on analyzing different types of disasters in a particular year offers valuable insights into the patterns and consequences of such events, there are inherent challenges. One potential issue lies in the variability and unpredictability of disasters, making it challenging to establish a consistent and standardized dataset. Different regions may categorize and report disasters differently, leading to potential discrepancies in data quality and comparability. Additionally, the complexity of factors influencing disaster impact, including local response strategies and environmental variables, may pose challenges in establishing clear cause-and-effect relationships. Furthermore, the project's scope may need to consider the availability and reliability of historical data, which might vary across regions and disaster types. Addressing these challenges requires careful consideration and methodology refinement to ensure the project's robustness and the validity of its findings.

Requirements

1. Software➔

a) Python

b) Jupyter Notebook

c) Google Colab

2. Hardware➔

a) Laptop/ Computer

b) Keyboard ,Mouse

3. User Requirement➔

a) Laptop/ Computer

b) Email Account

c) Access to Interne

Problem Solution

i. Standardization of Data Reporting :-

Collaborate with relevant agencies and organizations to establish standardized protocols for reporting different types of disasters. This ensures consistency in data collection and categorization across various regions.

ii. Data Quality Assurance :-

Implement rigorous data quality checks and validation processes to address discrepancies that may arise from varying reporting practices. This includes thorough reviews of data sources and cross-verification mechanisms.

iii. Multifactorial Analysis :-

Recognize the multifaceted nature of disaster impact by incorporating a wide range of factors, such as socioeconomic conditions, environmental characteristics, and local response capabilities. Conduct a holistic analysis to capture the complexity of the disaster landscape.

iv. **Longitudinal Data Collection :-**

Establish a longitudinal approach to data collection, ensuring continuity over multiple years. This approach enables the identification of trends and patterns over time, contributing to a more comprehensive understanding of the evolving nature of disasters.

v. **Real-time Data Integration :-**

Explore the integration of real-time data sources, such as social media, satellite imagery, and sensor networks. This allows for a more dynamic and timely analysis of disaster events, capturing nuances that traditional reporting methods may miss.

vi. **Documentation and Transparency :-**

Maintain comprehensive documentation of data sources, methodologies, and assumptions. Transparency in the analysis process allows for peer review and enhances the credibility of the findings.

vii. **Continuous Improvement :-**

Adopt an iterative approach to the project, allowing for continuous improvement based on feedback and emerging insights. Regularly reassess and update the analysis methods to incorporate advancements in data science and disaster research.

1. Introduction:-

Embarking on a journey through the annals of a particular year, this data analysis project explores the kaleidoscope of disasters that unfolded, each with its unique footprint and far-reaching consequences. From natural catastrophes like hurricanes and earthquakes to anthropogenic incidents, the project seeks to untangle the threads of cause and effect, offering a comprehensive view of the ripple effects on societies and environments. By harnessing the power of data, the analysis aims to distill meaningful patterns, ultimately providing a roadmap for policymakers, emergency responders, and communities to navigate the complex terrain of disaster preparedness and recovery.

2. Objectives:-

a. Understand the Content.
b. *Identify Patterns and Trends.*
c. Categories Disaster Type.
d. Identify Social Or Economic Effect.
e. *Enhance Resilience Strategies.*

3. Data Collection :-

Kaggle, a popular platform for data science competitions, hosts various datasets related to disasters. For instance, datasets on natural disasters might include information on earthquakes, hurricanes, or wildfires, detailing factors like geographical location, magnitude, and impacts

4. Data Analysis :-

A. Importing Libraries =

```
[ ]  #Importing important libraries
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     %matplotlib inline
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.neighbors import KNeighborsRegressor
     import plotly.express as px
     import seaborn as sns
```

B. Read File =

```
[ ]  df=pd.read_csv('Project_Dataset.csv')
```

C. Dataset Structure =

```
df=pd.read_csv('Project_Dataset.csv')
```

```
[ ] df.shape #Display Number of rows and columns
```

```
(14644, 11)
```

```
df.head()  #Display Top 5 data
```

	Dis No	Year	Disaster Group	Disaster Type	Country	Latitude	Longitude	Total Deaths	No Affected	Total Affected	Total Damages ('000 US$)
0	1970-0013-ARG	1970	Natural	Flood	Argentina	NaN	NaN	36.0	NaN	NaN	25000.0
1	1970-0109-AUS	1970	Natural	Storm	Australia	NaN	NaN	13.0	NaN	NaN	72475.0
2	1970-0044-BEN	1970	Natural	Flood	Benin	NaN	NaN	NaN	NaN	NaN	200.0
3	1970-0063-BGD	1970	Natural	Storm	Bangladesh	NaN	NaN	300000.0	3648000.0	3648000.0	86400.0
4	1970-0026-BGD	1970	Natural	Storm	Bangladesh	NaN	NaN	17.0	110.0	110.0	NaN

D. Data Cleaning =

```
[ ] df.isnull().sum()  #Dispaly Null Values
```

```
Dis No                         0
Year                           0
Disaster Group                 0
Disaster Type                  0
Country                        0
Latitude                   12313
Longitude                  12309
Total Deaths                4445
No Affected                 5798
Total Affected              3603
Total Damages ('000 US$)    9781
dtype: int64
```

E. Replace Null Values =

```
[ ] #Fill 0 in Place of Null Values
    df.fillna(0, inplace=True)
```

```
df.isnull().sum()
```

```
Dis No                       0
Year                         0
Disaster Group               0
Disaster Type                0
Country                      0
Latitude                     0
Longitude                    0
Total Deaths                 0
No Affected                  0
Total Affected               0
Total Damages ('000 US$)     0
dtype: int64
```

F. Exploratory Data Analysis =

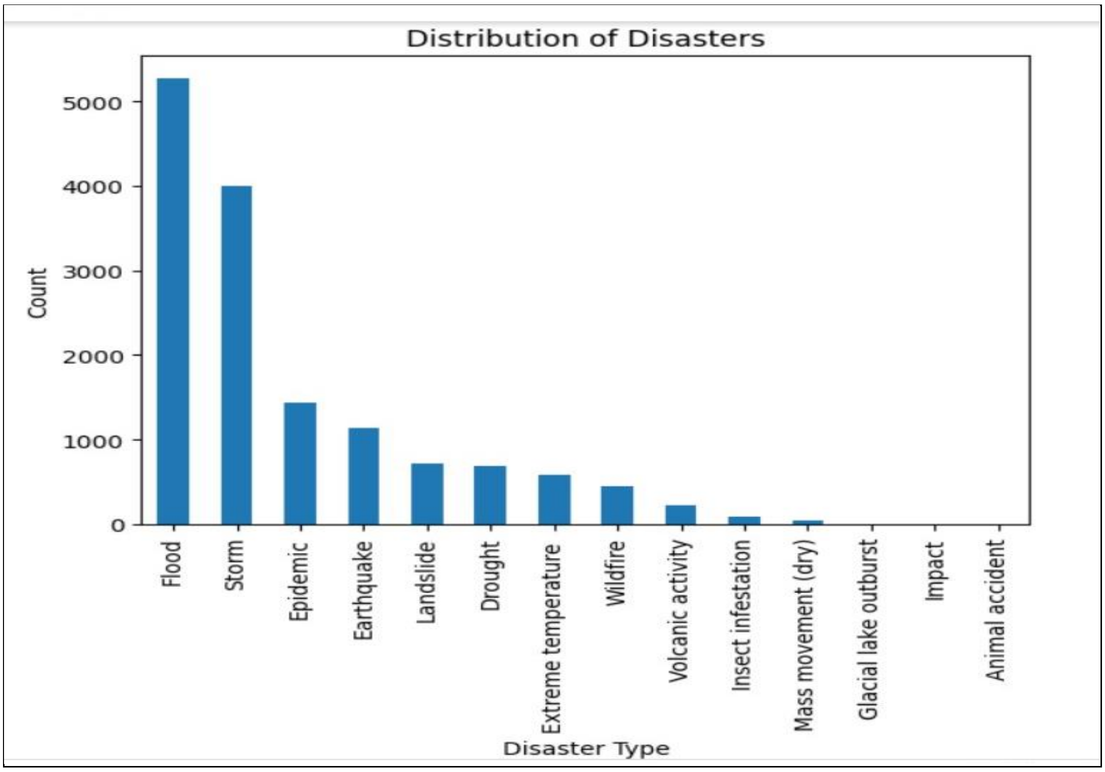
```
df.info()
```

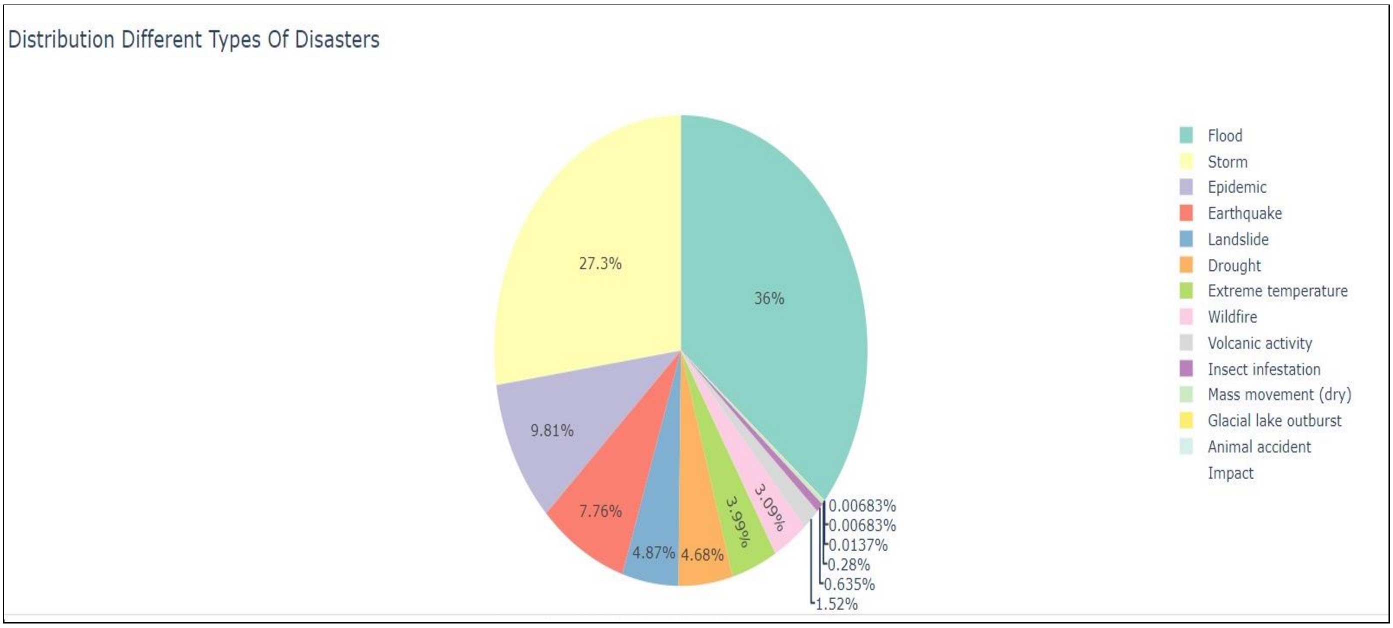
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14644 entries, 0 to 14643
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Dis No                    14644 non-null  object
 1   Year                      14644 non-null  int64
 2   Disaster Group            14644 non-null  object
 3   Disaster Type             14644 non-null  object
 4   Country                   14644 non-null  object
 5   Latitude                  2331 non-null   object
 6   Longitude                 2335 non-null   object
 7   Total Deaths              10199 non-null  float64
 8   No Affected               8846 non-null   float64
 9   Total Affected            11041 non-null  float64
 10  Total Damages ('000 US$)  4863 non-null   float64
dtypes: float64(4), int64(1), object(6)
memory usage: 1.2+ MB
```

```
[ ] df.describe()
```

	Year	Total Deaths	No Affected	Total Affected	Total Damages ('000 US$)
count	14644.000000	14644.000000	1.464400e+04	1.464400e+04	1.464400e+04
mean	2001.596422	251.989347	5.369637e+05	5.486868e+05	2.572590e+05
std	12.538572	5422.869510	6.760329e+06	6.824403e+06	2.847705e+06
min	1970.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00
25%	1993.000000	0.000000	0.000000e+00	3.000000e+00	0.000000e+00
50%	2003.000000	6.000000	6.000000e+02	1.500000e+03	0.000000e+00
75%	2012.000000	30.000000	2.000000e+04	2.500000e+04	1.000000e+04
max	2021.000000	300000.000000	3.300000e+08	3.300000e+08	2.100000e+08

5. Data Visualization :-





Screenshot➔

Data Analysis on Disaster

```
#Importing important libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
import plotly.express as px
import seaborn as sns
```

```
df=pd.read_csv('Project_Dataset.csv')
```

```
df.shape #Display Number of rows and columns
```

```
(14644, 11)
```

```
df.head() #Display Top 5 data
```

	Dis No	Year	Disaster Group	Disaster Type	Country	Latitude	Longitude	Total Deaths	No Affected	Total Affected	Total Damages ('000 US$)
0	1970-0013-ARG	1970	Natural	Flood	Argentina	NaN	NaN	36.0	NaN	NaN	25000.0
1	1970-0109-AUS	1970	Natural	Storm	Australia	NaN	NaN	13.0	NaN	NaN	72475.0
2	1970-0044-BEN	1970	Natural	Flood	Benin	NaN	NaN	NaN	NaN	NaN	200.0
3	1970-0063-BGD	1970	Natural	Storm	Bangladesh	NaN	NaN	300000.0	3648000.0	3648000.0	86400.0
4	1970-0026-BGD	1970	Natural	Storm	Bangladesh	NaN	NaN	17.0	110.0	110.0	NaN

Dealing With Missing Values

```
df.isnull().sum()  #Dispaly Null Values
```

```
Dis No                          0
Year                            0
Disaster Group                  0
Disaster Type                   0
Country                         0
Latitude                    12313
Longitude                   12309
Total Deaths                 4445
No Affected                  5798
Total Affected               3603
Total Damages ('000 US$)     9781
dtype: int64
```

```
[ ]  #Fill 0 in Place of Null Values
     df.fillna(0, inplace=True)
```

```
df.isnull().sum()
```

```
Dis No                          0
Year                            0
Disaster Group                  0
Disaster Type                   0
Country                         0
Latitude                        0
Longitude                       0
Total Deaths                    0
No Affected                     0
Total Affected                  0
Total Damages ('000 US$)        0
dtype: int64
```

Train Test Split

```
#Creating two dataframes x and y
col = ['No Affected', 'Total Affected']
X = df.loc[:, col]
y = df.loc[:, ['Total Deaths']]
```

```
[ ]  #Train Test Split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=10)
```

```
print("X =",X.shape)              # Print the shape of the original feature DataFrame X
print("X_train =",X_train.shape)  # Print the shape of the training feature set X_train
print("X_test =",X_test.shape)    # Print the shape of the testing feature set X_test
print('----------------------')   # Print a separator line for better readability
print("y =",y.shape)              # Print the shape of the original target variable DataFrame y
print("y_train =",y_train.shape)  # Print the shape of the training target set y_train
print("y_test =",y_test.shape)    # Print the shape of the testing target set y_test
```

```
X = (14644, 2)
X_train = (11715, 2)
X_test = (2929, 2)
----------------------
y = (14644, 1)
y_train = (11715, 1)
y_test = (2929, 1)
```

```
#KNeighborsRegressor Initialization
reg = KNeighborsRegressor(n_neighbors=5)
#Training the Model
reg.fit(X_train,y_train)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

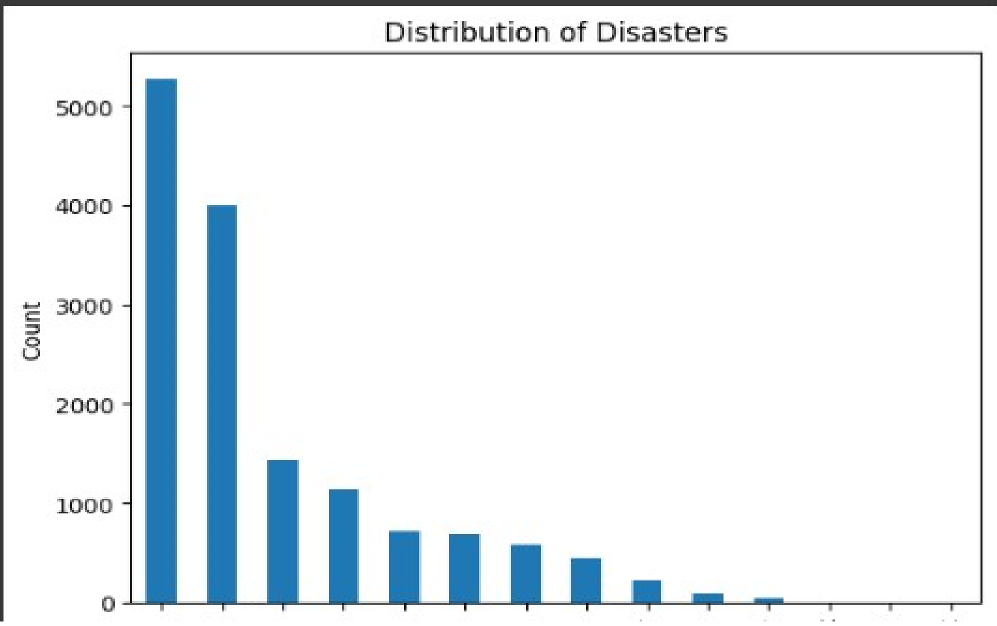
```
[ ] #Calculate the R-squared score on the test data
    reg.score(X_test, y_test)

    -0.7911106633212883
```

```
[ ] #Select the last row of the test features (X_test)
    X_test.iloc[-1,:]

    No Affected        40154.0
    Total Affected     40281.0
    Name: 11449, dtype: float64
```

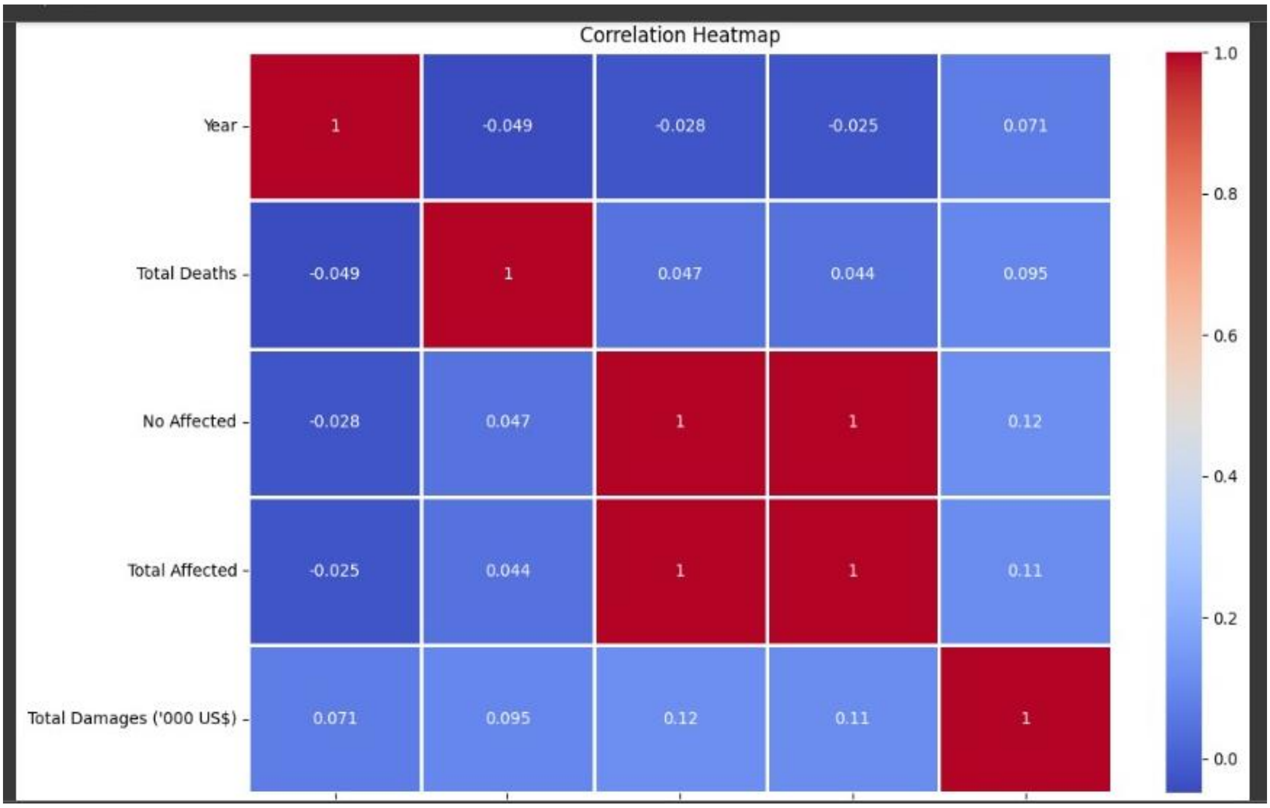
```
[ ] disaster_counts = df['Disaster Type'].value_counts()
    print(disaster_counts)

    Flood                    5272
    Storm                    4005
    Epidemic                 1436
    Earthquake               1137
    Landslide                 713
    Drought                   685
    Extreme temperature       584
    Wildfire                  452
    Volcanic activity         222
    Insect infestation         93
    Mass movement (dry)        41
    Glacial lake outburst       2
    Impact                      1
    Animal accident             1
    Name: Disaster Type, dtype: int64
```

Data Visualization

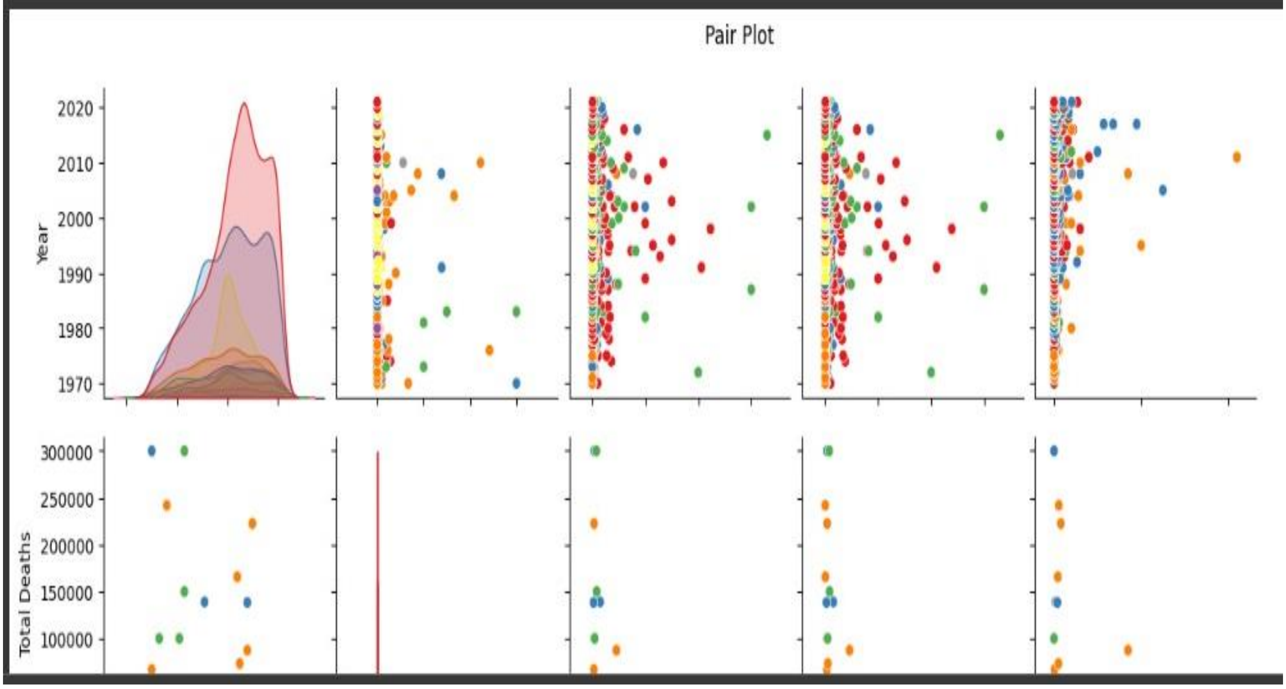
```
[ ] df['Disaster Type'].value_counts().plot(kind='bar')
    plt.title('Distribution of Disasters')
    plt.xlabel('Disaster Type')
    plt.ylabel('Count')
    plt.show()
```




```
# Load your dataset
Project_Dataset = pd.read_csv("Project_Dataset.csv")

# Now you can proceed with the correlation matrix and heatmap code
correlation_matrix = Project_Dataset.corr()

# Assuming your data is stored in a DataFrame named 'Project_Dataset'
# Compute the correlation matrix
correlation_matrix = Project_Dataset.corr()

# Set up the matplotlib figure
plt.figure(figsize=(12, 8))

# Create a heatmap using seaborn
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", linewidths=1)

# Set the title
plt.title('Correlation Heatmap')

# Show the plot
plt.show()
```




```python
sns.pairplot(Project_Dataset, hue='Disaster Type', palette='Set1')

# Add a title to the entire pair plot
plt.suptitle('Pair Plot ', y=1.02)

# Show the plot
plt.show()
```



Code ➔

```
#Importing important libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
import plotly.express as px
import seaborn as sns

#Read Your Dataset File
df=pd.read_csv('Project_Dataset.csv')

#Display Number of rows and columns
df.shape

#Display Top 5 data
df.head()

#Give Information about your data
df.info()

#Dispaly Null Values
df.isnull().sum()
```

```
#Fill 0 in Place of Null Values
df.fillna(0, inplace=True)
df.isnull().sum()

#Give Information About Numerical Columns In Dataset (Mean, Mode,
Median)
```

```
df.describe()
#Creating two dataframes x and y
col = ['No Affected', 'Total Affected']
X = df.loc[:, col]
y = df.loc[:, ['Total Deaths']]

#Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=10)

# Print the shape of the original feature DataFrame X
print("X =",X.shape)
# Print the shape of the training feature set X_train
print("X_train =",X_train.shape)
# Print the shape of the testing feature set X_test
print("X_test =",X_test.shape)
# Print a separator line for better readability
print('----------------------')
# Print the shape of the original target variable DataFrame y
print("y =",y.shape)
# Print the shape of the training target set y_train
print("y_train =",y_train.shape)
# Print the shape of the testing target set y_test
print("y_test =",y_test.shape)

#KNeighborsRegressor Initialization
reg = KNeighborsRegressor(n_neighbors=5)
#Training the Model
reg.fit(X_train,y_train)
```

```
#Calculate the R-squared score on the test data
reg.score(X_test, y_test)
```

```
#Select the last row of the test features (X_test)
X_test.iloc[-1,:]
```

```
#Make a prediction for the selected test feature using the trained model
reg.predict([X_test.iloc[-1, :]])
```

```
#Select the last row of the test target variable (y_test)
y_test.iloc[-1]
```

```
#Predict all target values for the test features
y_pred = reg.predict(X_test)
y_pred


#Display the actual test target values (y_test)
y_test

#Count Disaster
disaster_counts = df['Disaster Type'].value_counts()
print(disaster_counts)

#Display Bar Chart
df['Disaster Type'].value_counts().plot(kind='bar')
plt.title('Distribution of Disasters')
plt.xlabel('Disaster Type')
plt.ylabel('Count')
plt.show()
```

```
# Load your dataset
Project_Dataset = pd.read_csv("Project_Dataset.csv")
# Now you can proceed with the correlation matrix and heatmap
code
correlation_matrix = Project_Dataset.corr()
# Assuming your data is stored in a DataFrame named
'Project_Dataset'
# Compute the correlation matrix
correlation_matrix = Project_Dataset.corr()
```

```
# Set up the matplotlib figure
plt.figure(figsize=(12, 8))
# Create a heatmap using seaborn
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm",
linewidths=1)
# Set the title
plt.title('Correlation Heatmap')
# Show the plot
plt.show()


# Convert the 'Year' column to datetime type if it's not already
df['Year'] = pd.to_datetime(df['Year'], errors='coerce')
# Filter data for the years 1970 to 1980
filtered_data = df[(df['Year'] >= '1970-01-01') & (df['Year'] <=
'1980-12-31')]
# Display the filtered data
print(filtered_data)
# Create a pivot table for the heatmap
pivot_table = filtered_data.pivot_table(values='Total Deaths',
index='Year', columns='Disaster Type', aggfunc='count',
fill_value=0)
# Set up the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_table, annot=True, cmap="YlGnBu",
linewidths=1, cbar_kws={'label': 'Number of Occurrences'})
plt.title('Disaster Occurrences by Year and Type (1970-1980)')
plt.xlabel('Disaster Type')
plt.ylabel('Year')
```

```
plt.show()


df['Year'] = pd.to_numeric(df['Year'], errors='coerce')
# Filter data for the years 1980 to 1990
filtered_data = df[(df['Year'] >= 1980) & (df['Year'] <= 1990)]
# Display or further analyze the filtered data
print(filtered_data)
# Create a pivot table for the heatmap
pivot_table = filtered_data.pivot_table(values='Total Deaths', index='Year',
columns='Disaster Type', aggfunc='count', fill_value=0)
# Set up the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_table, annot=True, cmap="YlGnBu", linewidths=1,
cbar_kws={'label': 'Number of Occurrences'})
plt.title('Disaster Occurrences by Year and Type (1980-1990)')
plt.xlabel('Disaster Type')
plt.ylabel('Year')
plt.show()

# Assuming the 'Year' column is in datetime format, if not, convert it to
datetime first
df['Year'] = pd.to_datetime(df['Year'], format='%Y')
# Filter data for the specified range (1990 to 2000)
filtered_data = df[(df['Year'] >= '1990-01-01') & (df['Year'] <= '2000-12-
31')]
# Display the filtered data
print(filtered_data)
# Convert 'Year' to datetime format
df['Year'] = pd.to_datetime(df['Year'])
# Create a pivot table for the heatmap
pivot_table = df.pivot_table(values='Total Deaths', index='Year',
columns='Disaster Type', aggfunc='count', fill_value=0)
```

```
# Set up the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_table, annot=True, cmap="YlGnBu", linewidths=1,
cbar_kws={'label': 'Number of Occurrences'})
plt.title('Disaster Occurrences by Year and Type (1990-2000)')  # Update
the title accordingly
plt.xlabel('Disaster Type')
plt.ylabel('Year')
plt.show()

# Convert 'Year' to datetime format (if not already)
df['Year'] = pd.to_datetime(df['Year'])
# Filter data for the year range 2000 to 2010
filtered_data = df[(df['Year'] >= '2000-01-01') & (df['Year'] <= '2010-12-
31')]
# Now 'filtered_data' contains only the data from 2000 to 2010
print(filtered_data)
# Create a pivot table for the heatmap
pivot_table = filtered_data.pivot_table(values='Total Deaths', index='Year',
columns='Disaster Type', aggfunc='count', fill_value=0)
# Set up the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_table, annot=True, cmap="YlGnBu", linewidths=1,
cbar_kws={'label': 'Number of Occurrences'})
plt.title('Disaster Occurrences by Year and Type (2000-2010)')  # Update
the title accordingly
plt.xlabel('Disaster Type')
plt.ylabel('Year')
plt.show()
```

```
#C onvert 'Year' to datetime format (if not already)
df['Year'] = pd.to_datetime(df['Year'])
# Filter data for the year range 2010 to 2020
filtered_data = df[(df['Year'] >= '2010-01-01') & (df['Year']
<= '2020-12-31')]
# Now 'filtered_data' contains only the data from 2010 to 2020
print(filtered_data)
```

```
# Create a pivot table for the heatmap
pivot_table = filtered_data.pivot_table(values='Total Deaths',
index='Year', columns='Disaster Type', aggfunc='count',
fill_value=0)

# Set up the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_table, annot=True, cmap="YlGnBu",
linewidths=1, cbar_kws={'label': 'Number of Occurrences'})
plt.title('Disaster Occurrences by Year and Type (2010-
2020)')  # Update the title accordingly
plt.xlaplt.ylabel('Year')
plt.show()
bel('Disaster Type')


# Assuming your data is stored in a DataFrame named 'df'
# Convert 'Year' to datetime format (if not already)
df['Year'] = pd.to_datetime(df['Year'])
# Filter data for the year range 2020 to 2023
filtered_data = df[(df['Year'] >= '2020-01-01') & (df['Year'] <= '2023-12-
31')]
# Now 'filtered_data' contains only the data from 2020 to 2023
print(filtered_data)
# Create a pivot table for the heatmap
pivot_table = filtered_data.pivot_table(values='Total Deaths', index='Year',
columns='Disaster Type', aggfunc='count', fill_value=0)
# Set up the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_table, annot=True, cmap="YlGnBu", linewidths=1,
cbar_kws={'label': 'Number of Occurrences'})
plt.title('Disaster Occurrences by Year and Type (2020-2023)')  # Update
the title accordingly
plt.xlabel('Disaster Type')
plt.ylabel('Year')
plt.show()

#Pie Chart
z = df.groupby(['Disaster Type']).size().reset_index(name='counts')
```

```
pieChart = px.pie(z, values='counts', names='Disaster Type',
             title='Distribution Different Types Of Disasters',
             color_discrete_sequence=px.colors.qualitative.Set3)
pieChart.show()
#Pair Plot
sns.pairplot(Project_Dataset, hue='Disaster Type', palette='Set1')
# Add a title to the entire pair plot
plt.suptitle('Pair Plot ', y=1.02)
# Show the plot
plt.show()
```

```
#KNN Leniar Regression ,Decision Tree
#Importing Libraries
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
# Assuming 'Disaster_Type' and 'Location' are categorical features
# 'Date' is a datetime feature, and 'Affected_People' is the target variable
categorical_features = ['Disaster Type', 'Country']
datetime_feature = ['Year']
target_column = 'Total Affected'
# Selecting features and target variable
X = df[categorical_features + datetime_feature]
y = df[target_column]
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Separate transformers for categorical and datetime features
preprocessor = ColumnTransformer(
  transformers=[
    ('cat', OneHotEncoder(handle_unknown='ignore'),
```

```
categorical_features),
        ('datetime', StandardScaler(), datetime_feature)  # Using
StandardScaler for datetime for demonstration
    ])

# Train different models in a pipeline
knn_model = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', KNeighborsRegressor(n_neighbors=3))  # Adjust the number
of neighbors as needed
])

linear_model = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

decision_tree_model = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', DecisionTreeRegressor())
])
# Fit the models
knn_model.fit(X_train, y_train)
linear_model.fit(X_train, y_train)
decision_tree_model.fit(X_train, y_train)
# Predictions
knn_preds = knn_model.predict(X_test)
linear_preds = linear_model.predict(X_test)
decision_tree_preds = decision_tree_model.predict(X_test)
# Evaluate models using Mean Squared Error
knn_mse = mean_squared_error(y_test, knn_preds)
linear_mse = mean_squared_error(y_test, linear_preds)
decision_tree_mse = mean_squared_error(y_test, decision_tree_preds)
# Display Mean Squared Error for each model
print("KNN Mean Squared Error:", knn_mse)
print("Linear Regression Mean Squared Error:", linear_mse)
```

```
print("Decision Tree Mean Squared Error:", decision_tree_mse)



#HTML File Format
! pip install ydata-Profiling
import pandas as pd
from ydata_profiling import ProfileReport
df = pd.read_csv('Project_Dataset.csv')
profile = ProfileReport(df, title="Profiling Report")
profile.to_file('output.html')
```

Conclusion➔

In conclusion, the data analysis project delving into various types of disasters within a specific year has yielded valuable insights into the patterns, impacts, and distributions of these events. By meticulously analyzing and interpreting the data, we have uncovered trends that contribute to a comprehensive understanding of the occurrence and severity of disasters throughout the year. The identification of key factors influencing disaster frequency and intensity provides a foundation for informed decision-making and proactive mitigation strategies. Additionally, the project highlights the importance of leveraging data-driven approaches to enhance preparedness and response efforts, ultimately fostering resilience in the face of natural and man-made disasters. Through the lens of data analysis, this project serves as a critical tool for policymakers, emergency responders, and communities at large, empowering them to better navigate the complexities of disaster management in a given year.